UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/788,490	03/01/2004	Viera Bibr	T8468041US	9028

92030 7590 04/13/2011

Gowling Lafleur Henderson LLP
Suite 1600 1 First Canadian Place 100 King Street West
Toronto, ON M5X1G5
CANADA

EXAMINER	
WANG, BEN C	
ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
04/13/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/788,490	BIBR ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 January 2011*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-36 and 38-47* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-36 and 38-47* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated January 24, 2011, responding to the Office action mailed July 22, 2010 provided in the rejection of claims 1-36 and 38-47.

Claims 1-36 and 38-47 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are not persuasive. Please see the section of "Response to Arguments" below for details.

Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Response to Arguments

2. Applicant's arguments filed on January 24, 2011 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that, for examples:

(**A.1**) Applicant contends that Kougiouris does not disclose that *the mapping is present in the screen component* (Remarks on page 3, 2nd full paragraph).

Examiner Responses:

Examiner respectfully disagrees.

Firstly, Kougiouris teaches that the data management procedures may be managed by an executable component referred to as a 'dynamic interaction manager'. The markup language file *[interpreted as screen component]* may include a tag for instantiating the dynamic interaction manager, as appropriate to a particular markup language (Abstract, 2nd paragraph – emphasis added) and the dynamic interaction manager is operable to dynamically perform data management operations for GUI elements, based on the custom markup language attributes, and the dynamic interaction manager may interface with the application displaying the markup language GUI in order to receive dynamic programmatic events from the application (Abstract, 3rd paragraph. – emphasis added). Thus, the mapping configuration is located in the markup language file and the mapping process is triggered from the GUI application and is performed via the dynamic interaction manager which is just an executable component.

Secondly, Kougiouris further teaches the markup language file GUI descriptions may comprise information usable by the dynamic interaction manager to map GUI elements to data element (paragraph of [0055] – emphasis added)

Hence, Kougiouris indeed discloses the claim limitation recited above.

(**A.2**) Applicant contends that Kougiouris does not disclose that *binding one screen component to a data component and retrieving data from display on the GUI* and further *there is no automatic transfer of data to and from the GUI* (Remarks on page 3, 1st non-full paragraph).

Examiner Responses:

Examiner respectfully disagrees.

Firstly, Kougiouris teaches the application program 100 (see the Figure 1) may support the concept of data binding, wherein GUI elements are bound to a particular data elements (paragraph of [0049] - emphasis added)

Secondly, Kougiouris further teaches the dynamic interaction manager may enable this data binding by performing such operation as retrieving the data for a GUI element from a data source, transferring the data associated with a GUI element back to a data source (paragraph of [0050] - emphasis added)

Thirdly, Kougiouris discloses *data binding technology provides support for the automatic transfer of data to and from GUI element* (paragraph of [0009] - emphasis added)

Hence, Kougiouris indeed discloses the claim limitation recited above.

(**A.3**) Applicant contends that *"the claims require that there be a relationship between screen component and the data component and that that relationship is represented by a mapping defined by an identifier. As disclosed on page 11 line*

28 to page 12 line 30 [in the specification], *the relationship between the screen component and the data component is dynamic and that allow modifications. Thus, as disclosed, modifications made to either the screen component or the data component can be propagated to the other component. Applicants respectfully submit that the Kougiouris' data binding does not disclose the required relationship between the screen component and the data component. The GUI element and the data element are bound to one another regardless of what happens to the either. Kougiouris' binding does not permit this modification. Moreover, the claim requires that the mapping is present in the screen component. According to the Office Action, the mapping is disclosed by the dynamic interaction manager 104. As is evident by FIG. 1, the dynamic interaction manager 104 is a different element from the markup language file 102. As different elements, Kougiouris does not disclose that the mapping is present in the screen component* (Remarks on page 3, 1st non-full paragraph – with emphasis added).

Examiner Responses:

Examiner respectfully disagrees.

Firstly, Kougiouris indeed teaches *the mapping is present in the screen component* (see Examiner response for the first argument above)

Secondly, in response to Applicant's other arguments that the references fail to show certain features of Applicant's invention, it is noted that the features upon which Applicant relies (i.e., the relationship between the screen component

and the data component is dynamic and that allow modifications; the modifications made to either the screen component or the data component can be propagated to the other component as stated in the argument (A.3) with emphasis added) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See In re Van Geuns, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993)

Lastly, Kougiouris at least discloses a dynamic relationship between screen component and data component (e.g., [0009] - … data binding technology provides support for the automatic transfer of data to and from GUI elements … enable the various common data management operations to automatically be applied to data associated with a GUI element – emphasis added)

Examiner's Notes

3. Examiner cites particular columns and/or paragraphs and line numbers in the references as applied to claims below for the convenience of the applicant. Although the specified citations are representative of the teachings in the art and are applied to the specific limitations within the individual claim, other passages and figures may be applied as well. It is respectfully requested that, in preparing responses, the applicant fully consider the references in entirely as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior art or disclosed by the examiner.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-14, 18-29, 31, 36, and 38-42, and 44-47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hulai et al. (Pub. No. US 2003/0060896 A9) (hereinafter 'Hulai') in view of Warila et al. (Pub. No. US 2008/0313282 A1) (hereinafter 'Warila') and further in view of Kougiouris et al. (Pub. No. US 2004/0034833 A1) (hereinafter 'Kougiouris')

5. **As to claim 1** (Previously Presented), Hulai discloses a method for generating a screen element, based on a data object, of a component application executing on a wireless device for display on a user interface of a wireless device, the component application including a data component having at least one data field definition and a screen component having at least one screen element definition, the components being defined in a structured definition language, the method comprising the steps of:

- selecting the screen component corresponding to the screen element selected for display (e.g., Fig. 1, element 18 – User Interface; Fig. 2, element 67 – screen generation engine; Fig. 4, element 48 – User

Interface Definition Section; [0031], Lines 5-8; [0035], Lines 1-3; [0049], Lines 1-7; Fig. 8, element of S802 – i.e., create screen object; Fig. 9; Figs. 12-14; [0112], Lines 1-11; [0113], Lines 1-4); and

- obtaining a data object field value corresponding to the data field definition of the mapped data component (e.g., Fig. 16I, Sec. 3.2.3.3; [0039], Lines 1-7 – each of object classes includes attributes used to store parameters defined by the XML file, and functions allowing the XML entity to be processed at the mobile device)

Further, Hulai discloses providing the device an application definition file, containing definitions for a user interface format for the application at the wireless device (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *User Interface, Operating System and Architecture*, Warila discloses generating a screen element from the screen element definition *to include the data object field value* (e.g., In light of the specification, which stated '... *all changes to the data component can be immediately reflected on the screen and vice versa* (e.g., Hulai teaches the user's input data mapping/updating (*device-to-server direction*) from the screen component (*device*) to data component (*server*)*; this model allows building effective wireless applications based on server-to-device notifications; the data updates asynchronously pushed from the server are instantaneously reflected at the UI screen*.' (Abstract), *Warila* discloses '… a defined portion of the application can be addressed and updated in response to application events without

necessitating update of the entire application, the appearance and behavior of the application can be propagated with consistency across heterogeneous devices …' ([0032]), '… each editable area of the screen contains a value attribute that is updated automatically when the user operates the interface and changes information inside controls. these value attributes live within the superstructure itself …' ([0251]), [please also see Figure 4, elements 406 - Applications Data, 410 - Client Device, 404 - Server and Figure 18 – Current Value] - emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Warila into the Hulai's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai's system by taking, advancing and/or incorporating the Warila's system which offers significant advantages that the SimpleOS network model is that an application's code can be updated on the fly if so desired by the carrier or the developer as once suggested by Warila. (e.g., [0534])

Furthermore, Warila teaches a novel user interface, operating system, software language and architecture (e.g., Abstract) but Hulai and Warila do not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamic Interaction Manager for Markup Language Graphical User Interface*, Kougiouris discloses:

- identifying at least one mapping present in the screen component (**NOTE**: Firstly, Kougiouris teaches that the data management

procedures may be managed by an executable component referred to as a 'dynamic interaction manager'. The markup language file *[interpreted as screen component]* may include a tag for instantiating the dynamic interaction manager, as appropriate to a particular markup language (Abstract, 2nd paragraph – emphasis added) and the dynamic interaction manager is operable to dynamically perform data management operations for GUI elements, based on the custom markup language attributes, and the dynamic interaction manager may interface with the application displaying the markup language GUI in order to receive dynamic programmatic events from the application (Abstract, 3rd paragraph. – emphasis added). Thus, the mapping configuration is located in the markup language file and the mapping process is triggered from the GUI application and is performed via the dynamic interaction manager which is just an executable component. Secondly, Kougiouris further teaches the markup language file GUI descriptions may comprise information usable by the dynamic interaction manager to map GUI elements to data element (paragraph of [0055] – emphasis added), the mapping for specifying a relationship between the screen component and the data component as defined by an identifier representing the mapping (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D

[*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added);

- selecting the data component mapped by the mapping according to the mapping identifier (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added); and
- generating a screen element from the screen element definition according to the format of the data field definition as defined in the mapped data component (e.g., Fig. 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kougiouris into the Hulai-Warila's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai-Warila's system by taking, advancing and/or incorporating the Kougiouris's system which offers significant advantages that data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data and also enable the various common data management operations to automatically be applied to data associated with a GUI element as once suggested by Kougiouris (e.g., [0009] – emphasis added)

6. **As to claim 2** (Original) (incorporating the rejection in claim 1), Kougiouris discloses the method and the system wherein a plurality of the data field definitions of the data component is shared between the screen component and the data component as represented by the mapping (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data

element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added)

7. **As to claim 3** (Original) (incorporating the rejection in claim 2), Kougiouris discloses the method further comprising the step of linking the plurality of data field definitions to corresponding ones of the screen element definitions of the screen component as represented by the identifier (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

8. **As to claim 4** (Original) (incorporating the rejection in claim 2), Hulai discloses the method further comprising the step of detecting a user event of the user interface related to the screen element (e.g., Fig. 9, elements of S918, S920, S922, S924; [0096], Lines 1-13; Fig. 10; [0101]-[0103])

9. **As to claim 5** (Original) (incorporating the rejection in claim 4), Kougiouris discloses the method further comprising the step of identifying the mapping in the

screen component corresponding to the linked data component of the affected screen element (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added)

10. **As to claim 6** (Original) (incorporating the rejection in claim 5), Kougiouris discloses the method further comprising the step of updating the data object in a memory using the data field definition of the linked data component (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to

appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added)

11. **As to claim 7** (Original) (incorporating the rejection in claim 5), Hulai discloses the method further comprising the step of creating a new one of the data object in a memory using the data field definition of the linked data component (e.g., Fig. 2; [0035]-[0036] – object classes corresponding to XML entities supported by the virtual machine software, and possibly contained within an application definition file)

12. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 2, Hulai discloses the method and the system wherein the data object field value is obtained by being passed to the user interface as a screen parameter (e.g., [0039], Lines 1-7 – object classes define objects that allow device to process each of the supported XML entities at the mobile device; [0041], Lines 5-7 – at run time, instances of object classes corresponding to these classes are created

and populated with parameters contained within application definition file, as required; i.e., Fig. 16L, Sec. 3.3.3.5)

13. **As to claim 9** (Original) (incorporating the rejection in claim 2), Kougiouris discloses the method and the system wherein a first screen element definition is mapped by a first one of the identifiers to a first one of the data components and a second screen element definition is mapped by a second one of the identifiers to a second one of the data components different from the first data component (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

14. **As to claim 10** (Original) (incorporating the rejection in claim 9), Kougiouris discloses the method and the system wherein the first screen element definition and the second screen element definition are mapped to the same data component using the first identifier (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

15. **As to claims 11** (Original) (incorporating the rejection in claim 2), and Hulai discloses the method and the system wherein the structured definition language is XML based (e.g., Abstract, Lines 12-17)

16. **As to claim 12** (Original) (incorporating the rejection in claim 2), Kougiouris discloses the method and the system wherein the identifier is a simple primary key (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

17. **As to claim 13** (Original) (incorporating the rejection in claim 2), Kougiouris discloses the method wherein the identifier is a simple composite key (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

18. **As to claim 14** (Original) (incorporating the rejection in claim 2), Hulai discloses the method further comprising the step of receiving an asynchronous communication message by the device via a network coupled to the device, the message including a message data object (e.g., Fig. 1; Fig. 3; [0043]; Abstract, Lines 3-17; [0008] through [0011])

19. **As to claim 18** (Previously Presented), Hulai discloses a system for generating a screen element, based on a data object, of a component application executing on a wireless device, for display on a user interface of the wireless device, the component application including a data component having at least

one data field definition and a screen component having at least one screen element definition, the component being defined in a structured definition language, the system having memory for storing computer readable instructions and a processor configured to execute the instructions, the instructions for providing:

- a data manager for obtaining a data object field value corresponding to the data field definition of the mapped data component (e.g., Fig. 16l, Sec. 3.2.3.3; [0039], Lines 1-7 – each of object classes includes attributes used to store parameters defined by the XML file, and functions allowing the XML entity to be processed at the mobile device)

Further, Hulai discloses providing the device an application definition file, containing definitions for a user interface format for the application at the wireless device (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *User Interface, Operating System and Architecture*, Warila discloses for generating a screen element from the screen element definition to include the data object field value (e.g., In light of the specification, which stated '... *all changes to the data component can be immediately reflected on the screen and vice versa* (e.g., Hulai teaches the user's input data mapping/updating (*device-to-server direction*) from the screen component (*device*) to data component (*server*)*; this model allows building effective wireless applications based on server-to-device notifications; the data updates asynchronously pushed from the server are instantaneously reflected at*

the UI screen.' (Abstract), *Warila* discloses '… a defined portion of the application can be addressed and updated in response to application events without necessitating update of the entire application, the appearance and behavior of the application can be propagated with consistency across heterogeneous devices …' ([0032]), '… each editable area of the screen contains a value attribute that is updated automatically when the user operates the interface and changes information inside controls. these value attributes live within the superstructure itself …' ([0251]), [please also see Figure 4, elements 406 - Applications Data, 410 - Client Device, 404 - Server and Figure 18 – Current Value] - emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Warila into the Hulai's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai's system by taking, advancing and/or incorporating the Warila's system which offers significant advantages that the SimpleOS network model is that an application's code can be updated on the fly if so desired by the carrier or the developer as once suggested by Warila. (e.g., [0534)

Furthermore, Warila teaches a novel user interface, operating system, software language and architecture (e.g., Abstract) but Hulai and Warila do not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamic Interaction Manager for Markup Language Graphical User Interface*, Kougiouris discloses:

- a mapping manager for identifying at least one mapping present in the screen component (**NOTE**: Firstly, Kougiouris teaches that the data management procedures may be managed by an executable component referred to as a 'dynamic interaction manager'. The markup language file *[interpreted as screen component]* may include a tag for instantiating the dynamic interaction manager, as appropriate to a particular markup language (Abstract, 2nd paragraph – emphasis added) and the dynamic interaction manager is operable to dynamically perform data management operations for GUI elements, based on the custom markup language attributes, and the dynamic interaction manager may interface with the application displaying the markup language GUI in order to receive dynamic programmatic events from the application (Abstract, 3rd paragraph. – emphasis added). Thus, the mapping configuration is located in the markup language file and the mapping process is triggered from the GUI application and is performed via the dynamic interaction manager which is just an executable component.
  Secondly, Kougiouris further teaches the markup language file GUI descriptions may comprise information usable by the dynamic interaction manager to map GUI elements to data element (paragraph of [0055] – emphasis added), the mapping for specifying a relationship

between the screen component and the data component as defined by an identifier representing the mapping (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added), and for selecting the data component mapped by the mapping according to the mapping identifier (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added); and

- a presentation manager for generating a screen element from the screen element definition according to the format of the data field definition as defined in the mapped data component (e.g., Fig. 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kougiouris into the Hulai-Warila's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai-Warila's system by taking, advancing and/or incorporating the Kougiouris's system which offers significant advantages that data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data and also enable the various common data management operations to automatically be applied to data associated with a GUI element as once suggested by Kougiouris (e.g., [0009] – emphasis added)

20. **As to claim 19** (Original) (incorporating the rejection in claim 18), please refer to claim **13** as set forth accordingly.

21. **As to claim 20** (Original) (incorporating the rejection in claim 19), Kougiouris discloses the system wherein the plurality of data field definitions are linked to corresponding ones of the screen element definitions of the screen

component as represented by the identifier (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added)

22. **As to claim 21** (Previously Presented) (incorporating the rejection in claim 19), Hulai discloses the system is further comprising the presentation manager configured for detecting a user event of the user interface related to the screen element (e.g., Fig. 9, elements of S918, S920, S922, S924; [0096], Lines 1-13; Fig. 10; [0101] - [0103])

23. **As to claim 22** (Previously Presented) (incorporating the rejection in claim 21), Kougiouris discloses the system further comprising the mapping manager is further configured for identifying the mapping in the screen component

corresponding to the linked data component of the affected screen element (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added)

24. **As to claim 23** (Previously Presented) (incorporating the rejection in claim 22), Kougiouris discloses the system wherein the data manager is further configured for updating the data object in a memory using the data field definition of the linked data component (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

25. **As to claim 24** (Previously Presented) (incorporating the rejection in claim 22), Kougiouris discloses the system wherein the data manager is further configured for creating a new one of the data object in a memory using the data field definition of the linked data component (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added)

26. **As to claim 25** (Previously Presented) (incorporating the rejection in claim 19), please refer to claim **8** as set forth accordingly.

27. **As to claim 26** (Original) (incorporating the rejection in claim 19), please refer to claim **9** as set forth accordingly.

28. **As to claim 27** (Original) (incorporating the rejection in claim 26), please refer to claim **10** as set forth accordingly.

29. **As to claim 28** (Previously Presented) (incorporating the rejection in claim 19), please refer to claim **11** as set forth accordingly.

30. **As to claim 29** (Original) (incorporating the rejection in claim 19), please refer to claim **12** as set forth accordingly.

31. **As to claim 31** (Original) (incorporating the rejection in claim 19), Hulai discloses the system further comprising a communication manager for receiving

an asynchronous communication message by the device via a network coupled to the device, the message including a message data object (e.g., Fig. 1; Fig. 3; [0043]; Abstract, Lines 3-17; [0008] through [0011])

32. **As to claim 36** (Previously Presented), Hulai discloses a wireless device for generating a screen element, based on a data object, of a component application executing on the wireless device for display on a user interface of the wireless device, the component application including a data component having at least one data field definition and a screen component having at least one screen element definition, the component being defined in a structured definition language, the wireless device comprising the steps of:

- means for selecting the screen component corresponding to the screen element selected for display (e.g., Fig. 1, element 18 – User Interface; Fig. 2, element 67 – screen generation engine; Fig. 4, element 48 – User Interface Definition Section; [0031], Lines 5-8; [0035], Lines 1-3; [0049], Lines 1-7; Fig. 8, element of S802 – i.e., create screen object; Fig. 9; Figs. 12-14; [0112], Lines 1-11; [0113], Lines 1-4; [0049], Lines 4-7 – a user interface definition section, specific to the user interface for the device);
- means for selecting the data component mapped by the mapping (e.g., Fig. 4, element 52 – Device Local Data Definition Section;[0105], Lines 6-9; [0122], Lines 1-7; [0049], Lines 9-11 – a local data definition section defining the format of data to be stored locally on the mobile device by the application); and

- means for obtaining a data object field value corresponding to the data field definition of the mapped data component (e.g., Fig. 16l, Sec. 3.2.3.3; [0039], Lines 1-7 – each of object classes includes attributes used to store parameters defined by the XML file, and functions allowing the XML entity to be processed at the mobile device)

Further, Hulai discloses providing the device an application definition file, containing definitions for a user interface format for the application at the wireless device (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *User Interface, Operating System and Architecture*, Warila discloses means for generating a screen element from the screen element definition to include the data object field value (e.g., In light of the specification, which stated '... *all changes to the data component can be immediately reflected on the screen and vice versa* (e.g., Hulai teaches the user's input data mapping/updating (*device-to-server direction*) from the screen component (*device*) to data component (*server*))*; this model allows building effective wireless applications based on server-to-device notifications; the data updates asynchronously pushed from the server are instantaneously reflected at the UI screen*.' (Abstract), *Warila* discloses '… a defined portion of the application can be addressed and updated in response to application events without necessitating update of the entire application, the appearance and behavior of the application can be propagated with consistency across heterogeneous devices …' ([0032]), '… each editable area of the screen contains a value

attribute that is updated automatically when the user operates the interface and changes information inside controls. these value attributes live within the superstructure itself …' ([0251]), [please also see Figure 4, elements 406 - Applications Data, 410 - Client Device, 404 - Server and Figure 18 – Current Value] - emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Warila into the Hulai's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai's system by taking, advancing and/or incorporating the Warila's system which offers significant advantages that the SimpleOS network model is that an application's code can be updated on the fly if so desired by the carrier or the developer as once suggested by Warila. (e.g., [0534)

Furthermore, Warila teaches a novel user interface, operating system, software language and architecture (e.g., Abstract) but Hulai and Warila do not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamic Interaction Manager for Markup Language Graphical User Interface*, Kougiouris discloses:

- means for identifying at least one mapping present in the screen component (**NOTE**: Firstly, Kougiouris teaches that the data management procedures may be managed by an executable component referred to as a 'dynamic interaction manager'. The markup

language file *[interpreted as screen component]* may include a tag for instantiating the dynamic interaction manager, as appropriate to a particular markup language (Abstract, 2nd paragraph – emphasis added) and the dynamic interaction manager is operable to dynamically perform data management operations for GUI elements, based on the custom markup language attributes, and the dynamic interaction manager may interface with the application displaying the markup language GUI in order to receive dynamic programmatic events from the application (Abstract, 3rd paragraph. – emphasis added). Thus, the mapping configuration is located in the markup language file and the mapping process is triggered from the GUI application and is performed via the dynamic interaction manager which is just an executable component.

Secondly, Kougiouris further teaches the markup language file GUI descriptions may comprise information usable by the dynamic interaction manager to map GUI elements to data element (paragraph of [0055] – emphasis added), the mapping for specifying a relationship between the screen component and the data component (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008]

- … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added); and

- means for generating a screen element from the screen element definition according to the format of the data field definition as defined in the mapped data component (e.g., Fig. 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kougiouris into the Hulai-Warila's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai-Warila's system by taking, advancing and/or incorporating the Kougiouris's system which offers significant advantages that data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the

distinction between data and the markup language description of the GUI for displaying the data and also enable the various common data management operations to automatically be applied to data associated with a GUI element as once suggested by Kougiouris (e.g., [0009] – emphasis added)

33. **As to claim 38** (Previously Presented), Hulai discloses a computer readable medium comprising instructions for generating a screen element, based on a data object, of a component application executing on a wireless device for display on a user interface of the wireless device, the component application including a data component having at least one data field definition and a screen component having at least one screen element definition, the components being defined in a structured definition language, the instructions, when implemented on a computing device, cause the computing device, cause the computing device to implement the steps of:

- selecting the screen component corresponding to the screen element selected for display (e.g., Fig. 1, element 18 – User Interface; Fig. 2, element 67 – screen generation engine; Fig. 4, element 48 – User Interface Definition Section; [0031], Lines 5-8; [0035], Lines 1-3; [0049], Lines 1-7; Fig. 8, element of S802 – i.e., create screen object; Fig. 9; Figs. 12-14; [0112], Lines 1-11; [0113], Lines 1-4); and
- obtaining a data object field value corresponding to the data field definition of the mapped data component (e.g., Fig. 16l, Sec. 3.2.3.3; [0039], Lines 1-7 – each of object classes includes attributes used to store parameters

defined by the XML file, and functions allowing the XML entity to be processed at the mobile device)

Further, Hulai discloses providing the device an application definition file, containing definitions for a user interface format for the application at the wireless device (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *User Interface, Operating System and Architecture*, Warila discloses generating a screen element from the screen element definition to include the data object field value (e.g., In light of the specification, which stated '... *all changes to the data component can be immediately reflected on the screen and vice versa* (e.g., Hulai teaches the user's input data mapping/updating (*device-to-server direction*) from the screen component (*device*) to data component (*server*))*; this model allows building effective wireless applications based on server-to-device notifications; the data updates asynchronously pushed from the server are instantaneously reflected at the UI screen*.' (Abstract), *Warila* discloses '… a defined portion of the application can be addressed and updated in response to application events without necessitating update of the entire application, the appearance and behavior of the application can be propagated with consistency across heterogeneous devices …' ([0032]), '… each editable area of the screen contains a value attribute that is updated automatically when the user operates the interface and changes information inside controls. these value attributes live within the superstructure itself …' ([0251]), [please also see Figure 4, elements 406 -

Applications Data, 410 - Client Device, 404 - Server and Figure 18 – Current Value] - emphasis added)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Warila into the Hulai's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai's system by taking, advancing and/or incorporating the Warila's system which offers significant advantages that the SimpleOS network model is that an application's code can be updated on the fly if so desired by the carrier or the developer as once suggested by Warila. (e.g., [0534)

Furthermore, Warila teaches a novel user interface, operating system, software language and architecture (e.g., Abstract) but Hulai and Warila do not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamic Interaction Manager for Markup Language Graphical User Interface*, Kougiouris discloses:

- identifying at least one mapping present in the screen component (**NOTE**: Firstly, Kougiouris teaches that the data management procedures may be managed by an executable component referred to as a 'dynamic interaction manager'. The markup language file *[interpreted as screen component]* may include a tag for instantiating the dynamic interaction manager, as appropriate to a particular markup language (Abstract, 2nd paragraph – emphasis added) and the dynamic

interaction manager is operable to dynamically perform data management operations for GUI elements, based on the custom markup language attributes, and the dynamic interaction manager may interface with the application displaying the markup language GUI in order to receive dynamic programmatic events from the application (Abstract, 3rd paragraph. – emphasis added). Thus, the mapping configuration is located in the markup language file and the mapping process is triggered from the GUI application and is performed via the dynamic interaction manager which is just an executable component. Secondly, Kougiouris further teaches the markup language file GUI descriptions may comprise information usable by the dynamic interaction manager to map GUI elements to data element (paragraph of [0055] – emphasis added), the mapping for specifying a relationship between the screen component and the data component as defined by an identifier representing the mapping (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and

an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various common data management operations to automatically be applied to data associated with a GUI element - emphasis added);

- selecting the data component mapped by the mapping according to the mapping identifier (e.g., [0055] – [0056] - … for example, the HSDATASRC value may identify a particular database table, and the HSDATAFLD attribute may identify a column of the table [*interpreted as the mapping according to the mapping identifier*] … - emphasis added);
- generating a screen element from the screen element definition according to the format of the data field definition as defined in the mapped data component (e.g., . 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kougiouris into the Hulai-Warila's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai-Warila's system by taking, advancing and/or incorporating the Kougiouris's system which offers significant advantages that data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data and also enable the various common data management operations to automatically be applied to data associated with a GUI element as once suggested by Kougiouris (e.g., [0009] – emphasis added)

34. **As to claim 39** (Previously Presented) (incorporating the rejection in claim 1), Kougiouris discloses the method wherein the use of the mapping reduces the amount of instructions to define the screen component or perform screen handling (e.g., [0130] - … Providing these types of formatting/validating capabilities may be particularly important for applications ... in which information is shared across many applications, and where applications may expect information to be encoded or demarcated in particular ways – emphasis added)

35. **As to claim 40** (Previously Presented) (incorporating the rejection in claim 14), Kougiouris discloses the method further comprising dynamically defining said data field definition at runtime in response to a format of said message data object received in said message (e.g., Fig. 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

36. **As to claim 41** (Previously Presented) (incorporating the rejection in claim 18), please refer to claim **39** as set forth accordingly.

37. **As to claim 42** (Previously Presented) (incorporating the rejection in claim 31), please refer to claim **40** as set forth accordingly.

38. **As to claim 44** (Previously Presented) (incorporating the rejection in claim 36), please refer to claim **39** as set forth accordingly.

39. **As to claim 45** (Previously Presented) (incorporating the rejection in claim 36), Hulai discloses the wireless device wherein a plurality of the data field definitions of the data component is shared between the screen component and the data component as represented by the mapping; and wherein the wireless device further comprises:

- means for receiving an asynchronous communication message by the device via a network coupled to the device, the message including a message data object (e.g., Fig. 1; Fig. 3; [0043]; Abstract, Lines 3-17; [0008] through [0011]); and further, Hulai discloses means to dynamically define said data field definition at runtime in response to a format of said message data object received in said message (e.g., Fig. 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

40. **As to claim 46** (Previously Presented) (incorporating the rejection in claim 38), please refer to claim **39** as set forth accordingly.

41. **As to claim 47** (Previously Presented) (incorporating the rejection in claim 38), please refer to claim **45** as set forth accordingly.

42. Claims 35 and 43 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hulai in view of Kougiouris.

43. **As to claim 35** (Previously Presented), Hulai discloses a method for generating a data object of a component application executing on a wireless device based on a change in a screen element displayed on a user interface of a wireless device, the component application including a data component having at least one data field definition and a screen component having at least one screen element definition, the component being defined in a structured definition language, the method comprising the steps of:

- selecting the screen component corresponding to the screen element (e.g., Fig. 1, element 18 – User Interface; Fig. 2, element 67 – screen generation engine; Fig. 4, element 48 – User Interface Definition Section; [0031], Lines 5-8; [0035], Lines 1-3; [0049], Lines 1-7; Fig. 8, element of S802 – i.e., create screen object; Fig. 9; Figs. 12-14; [0112], Lines 1-11; [0113], Lines 1-4);
- selecting the data component mapped by the mapping (e.g., Fig. 4, element 52 – Device Local Data Definition Section;[0105], Lines 6-9; [0122], Lines 1-7; [0049], Lines 9-11 – a local data definition section

defining the format of data to be stored locally on the mobile device by the application); and

- obtaining a changed value from the screen element corresponding to the mapped data component (e.g., [0036] – parser may convert each XML tag contained in the application definition file, and its associated data to tokens, for later processing; Fig. 9; [0096], Lines 1-13; [0117], Lines 6-18)

Further Hulai discloses providing the device an application definition file, containing definitions for a user interface format for the application at the wireless device (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamic Interaction Manager for Markup Language Graphical User Interface*, Kougiouris discloses:

- identifying at least one mapping present in the screen component (**NOTE**: Firstly, Kougiouris teaches that the data management procedures may be managed by an executable component referred to as a 'dynamic interaction manager'. The markup language file *[interpreted as screen component]* may include a tag for instantiating the dynamic interaction manager, as appropriate to a particular markup language (Abstract, 2nd paragraph – emphasis added) and the dynamic interaction manager is operable to dynamically perform data management operations for GUI elements, based on the custom markup language attributes, and the dynamic interaction manager may interface with the application displaying the markup language GUI in

order to receive dynamic programmatic events from the application (Abstract, 3rd paragraph. – emphasis added). Thus, the mapping configuration is located in the markup language file and the mapping process is triggered from the GUI application and is performed via the dynamic interaction manager which is just an executable component. Secondly, Kougiouris further teaches the markup language file GUI descriptions may comprise information usable by the dynamic interaction manager to map GUI elements to data element (paragraph of [0055] – emphasis added), the mapping for specifying a relationship between the screen component and the data component (e.g., Fig. 1, markup language file 102 [*interpreted as screen component*]; dynamic interaction manager 104 [*interpreted as mapping mechanism/data binding – see further details below*]; data management component A/B/C/D [*interpreted as data component*]; Fig. 2, dynamic interaction manager binds GUI elements to appropriate data elements 203; [0008] - … data binding … refers to a relationship in which a GUI elements is bound to a data element. The data element for the GUI element may be specified in the markup language description for the GUI element, and an application may automatically perform the binding to the data element …; [0009] – Thus, data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data … enable the various

common data management operations to automatically be applied to data associated with a GUI element - emphasis added); and

- assigning the changed value to a data field value of the data object according to the format of the data field definition as defined in the mapped data component (e.g., Fig. 5, *IHSFormatter* 352; [0099] - Fig. 5 – Exemplary Embodiment of Validation/Formatting Interface; [0100] – [0106])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kougiouris into the Hulai's system to further provide other limitations stated above in the Hulai system.

The motivation is that it would further enhance the Hulai's system by taking, advancing and/or incorporating the Kougiouris's system which offers significant advantages that data binding technology provides support for the automatic transfer of data to and from GUI elements, helping to maintain the distinction between data and the markup language description of the GUI for displaying the data and also enable the various common data management operations to automatically be applied to data associated with a GUI element as once suggested by Kougiouris (e.g., [0009] – emphasis added)

44. **As to claim 43** (Previously Presented) (incorporating the rejection in claim 35), please refer to claim **39** as set forth accordingly.

45. Claims 15-17 and 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hulai in view of Warila and Kougiouris and further in view of Saulpaugh et al., (Pat. No. US 7,010,573 B1) (hereinafter 'Saulpaugh')

46. **As to claim 15** (Previously Presented) (incorporating the rejection in claim 14), Hulai discloses employing Virtual Machine and XML messaging technologies (e.g., Abstract, Lines 12-17), but Hulai, Warila and Kougiouris do not explicitly disclose the limitations stated below.

However, in an art of *message gates using a shared transport in a distributed computing environment*, Saulpaugh discloses checking the asynchronous communication message for the mapping corresponding to the data component of the application provisioned on the device (e.g., Col. 7, Lines 1-6 – the messages may be in a data representation language such as eXtensible Mark0up Languages (XML), 12-16 – each such message may be sent through a client message gate that may verify the correctness of the message)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Saulpaugh into the Hulai- Warila-Kougiouris's system to further provide the limitations stated above in the Hulai-Warila Kougiouris's system.

The motivation is that it would further enhance the Hulai-Warila-Kougiouris's system by taking, advancing and/or incorporating the Saulpaugh's system which offers significant advantages for providing a simple way to connect various types of intelligent devices to allow for communication and sharing of

resources while avoiding the interoperability and complex configuration problems existing in conventional networks as once suggested by Saulpaugh (e.g., Col. 2, Lines 3-7)

47. **As to claim 16** (Previously Presented) (incorporating the rejection in claim 15), Hulai discloses the method further comprising the step of updating the message data object corresponding to the message in a memory using the data field definition of the linked data component and then reflecting that data change in the screen element linked to the data object (e.g., [0085]; [0086] – the particular identity of the mobile device on which the application is to be presented may be identified by a suitable identifier, in the form of a header contained in the server side application output; [0097] – virtual machine software further maintains a list identifying each instance of each event and action object, and an associated identifier of an event; i.e., Fig. 16II, Sec. 6.6.3.2, Sec. 6.6.3.3, Sec. 6.6.3.4; Fig. 16JJ, Sec. 6.7.3.2; Fig. 9; [0096], Lines 16-19).

48. **As to claim 17** (Original) (incorporating the rejection in claim 15), Hulai discloses the method further comprising the step of creating the data object corresponding to the message in a memory using the data field definition of the linked data component ([0040], Lines 4-9; [0041], Lines 5-7; i.e., [0051]; Fig. 9; [0096], Lines 16-19)

49. **As to claim 32** (Previously Presented) (incorporating the rejection in claim 19), Saulpaugh discloses the system further comprising the mapping manager

configured for checking the message for the mapping corresponding to the data component of the application provisioned on the device (e.g., Col. 7, Lines 1-6 – the messages may be in a data representation language such as eXtensible Mark0up Languages (XML), 12-16 – each such message may be sent through a client message gate that may verify the correctness of the message)

50. **As to claim 33** (Previously Presented) (incorporating the rejection in claim 32), Hulai discloses the system further comprising the data manager configured for updating the message data object in a memory using the data field definition of the linked data component (e.g., [0085]; [0086] – the particular identity of the mobile device on which the application is to be presented may be identified by a suitable identifier, in the form of a header contained in the server side application output; [0097] – virtual machine software further maintains a list identifying each instance of each event and action object, and an associated identifier of an event; i.e., Fig. 16II, Sec. 6.6.3.2, Sec. 6.6.3.3, Sec. 6.6.3.4; Fig. 16JJ, Sec. 6.7.3.2; Fig. 9; [0096], Lines 16-19)

51. **As to claim 34** (Previously Presented) (incorporating the rejection in claim 32), Hulai discloses the system further comprising the data manager configured for creating the data object corresponding to the message in a memory using the data field definition of the linked data component (e.g., [0040], Lines 4-9; [0041], Lines 5-7; i.e., [0051]; Fig. 9; [0096], Lines 16-19)

52. Claims 13 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hulai in view of Warila and Kougiouris and further in view of Greene et al., (Pat. No. US 6,868,441 B2) (hereinafter 'Greene')

53. **As to claims 13** (Original) (incorporating the rejection in claim 2), Hulai discloses employing Virtual Machine and XML messaging technologies (e.g., Abstract, Lines 12-17), but Hulai, Warila and Kougiouris do not explicitly disclose the limitations stated below.

However, in an art of *method and system for implementing a global ecosystem of interrelated services*, Greene discloses the method and the system wherein the identifier is a composite key (e.g., Col. 69, Lines 1-10 – for example, the PK for a given entity might be a string or an integer, or it might be a composite key having more than one component).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Greene into the Hulai-Warila-Kougiouris's system to further provide the limitations stated above in the Hulai-Warila-Kougiouris's system.

The motivation is that it would further enhance the Hulai-Warila-Kougiouris's system by taking, advancing and/or incorporating the Greene's system which offers advantages for providing alternate, domain specific primary keys that can be used by the specific application, or by custom logic within the entity implementation, and checked for uniqueness by the central entity manager,

using for example, a hashing or directory service as once suggested by Greene (e.g., Col. 69, Lines 1-10)

54. **As to claim 30** (Original) (incorporating the rejection in claim 19), please refer to claim **13** as set forth accordingly.

Conclusion

55. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192

/Michael J. Yigdall/
Primary Examiner, Art Unit 2192